คีย์บอร์ดบัฟเฟอร์ ขนาดที่ใหญ่กว่า

ขยายขนาดบัฟเฟอร์ของคีย์บอร์ด ด้วยโปรแกรม BIG.COM เก็บคาแรคเตอร์มากขึ้น ช่วยคีย์ข้อมูลได้เร็ว

ถ้าคุณเป็นคนที่ใช้คีย์บอร์ดรวดเร็วมาก คุณมักจะพบกับปัญหาที่คีย์บอร์ดเต็มและมีเสียงเตือนจากดอสมาบ่อย ๆ โดยปกติคีย์บอร์ดบัฟเฟอร์จะมีขนาดเพียง 15 คาแรคเตอร์เท่านั้น พอเรากดคีย์บอร์ดเข้าไปก่อนที่คาแรคเตอร์จะถูกอ่านออกมาจากบัฟเฟอร์ ก็จะมีเสียงร้องเตือนบอกว่าบัฟเฟอร์เต็มแล้ว ดังนั้นถ้าเราขยายขนาดของคีย์บอร์ดบัฟเฟอร์นี้ให้ใหญ่ขึ้นได้ก็คงดีไม่น้อย ซึ่งก็คงจะหนีไม่พ้นที่จะต้องเขียนโปรแกรมขึ้นมาทำหน้าที่ขยายขนาดนี้ โดยผมได้ขยายให้คีย์บอร์ดบัฟเฟอร์เก็บได้ถึง 80 คาแรคเตอร์ หลังจากที่โหลดโปรแกรมฝังลงบนหน่วยความจำแล้ว โปรแกรมจะเริ่มทำงานทันทีที่คีย์บอร์ดบัฟเฟอร์เต็ม นอกจากนี้โปรแกรมจะแสดงคาแรคเตอร์ที่เก็บอยู่ในบัฟเฟอร์ใหม่นี้ด้วย

คีย์บอร์ดบัฟเฟอร์

ผมคงต้องขอพูดถึงรายละเอียดของคีย์บอร์ดบัฟเฟอร์สักเล็กน้อย ซึ่งหากท่านผู้อ่านทราบถึงเรื่องนี้แล้ว ก็ขอให้ข้ามไปเลยนะครับ

คีย์บอร์ดบัฟเฟอร์จะเป็นเนื้อที่ส่วนหนึ่งของหน่วยความจำบริเวณของไบออส โดยเริ่มที่เซกเมนต์ 40h และออฟเซต 1Eh ซึ่งจะจองเนื้อที่เอาไว้ 16 เวิร์ด สำหรับโครงสร้างของบัฟเฟอร์จะเป็นลิสต์แบบวงกลม (Circular list) นอกจากนี้ในไบออสจะมีตัวแปร HEAD, TAIL ซึ่งทำหน้าที่เป็นพอยน์เตอร์ชี้หัวและหางของลิสต์

เมื่อใดก็ตามที่คุณกดคีย์บอร์ด คาแรคเตอร์ที่เข้ามาใหม่ก็จะถูกเอาเข้ามาเก็บไว้ในบัฟเฟอร์ในตำแหน่งที่พอยน์เตอร์ TAIL ชี้อยู่ และเมื่อดอสอ่านคาแรคเตอร์ออกจากบัฟเฟอร์ ดอสจะอ่านตำแหน่งที่พอยน์เตอร์ HEAD ชี้อยู่ ทุกครั้งที่มีการอ่านและเขียนค่าลงในบัฟเฟอร์ ค่า HEAD และ TAIL จะถูกเพิ่มค่าขึ้นทีละหนึ่ง โดย HEAD จะมีค่าน้อยกว่า TAIL เสมอ

สำหรับค่าคาแรคเตอร์แต่ละตัวที่เก็บในบัฟเฟอร์ จะเก็บค่าสแกนโค้ดในไบต์สูงและค่ารหัสแอสกีในไบต์ต่ำ

การทำงานของโปรแกรม

โปรแกรม BIG.COM จะแยกออกได้เป็นสองส่วน คือ ส่วนแรกทำหน้าที่แทนอินเตอร์รัพต์หมายเลข 9 ส่วนที่สองทำหน้าที่แทนอินเตอร์รัพต์หมายเลข 1Ch โปรแกรมแสดงดังรูปที่ 1

สำหรับส่วนแรกจะคอยตรวจสอบทุกครั้งที่มีการกดคีย์บอร์ด โดยตรวจค่าคาแรคเตอร์ตัวสุดท้ายในคีย์บอร์ดบัฟเฟอร์ถูกอ่านเข้ามาเก็บในบัฟเฟอร์ของ BIG หรือยัง ซึ่งจะทำโดยเปรียบเทียบค่าในคีย์บอร์ดบัฟเฟอร์ที่ TAIL ชี้อยู่กับค่าของ BUF ถ้าค่าเท่ากันแสดงว่าอ่านเข้ามาแล้ว และจะต้องตรวจสอบค่า HEAD ว่าถูกเพิ่มค่าขึ้นเนื่องจากเป็นการกดคีย์ซ้ำกัน โดยเปรียบเทียบกับค่า OLD-HEAD นอกจากนี้ก็ต้องตรวจสอบกรณีที่คีย์ที่กดเข้ามาเป็น Back Space เพื่อจะลบคาแรคเตอร์ออกจากบัฟเฟอร์ของ BIG จากนั้น BIG จะย้ายเอาคาแรคเตอร์จากคีย์บอร์ดบัฟเฟอร์มาเก็บไว้ในบัฟเฟอร์ของตัวเองแทน แล้วนำคาแรคเตอร์ทั้งหมดในบัฟเฟอร์ของตัวเองออกมาแสดงบนจอภาพด้วย

ส่วนที่สองซึ่งจะทำหน้าที่แทนอินเตอร์รัพต์หมายเลข 1Ch นั้น โดยจะทำหน้าที่ย้ายเอาคาแรคเตอร์จากบัฟเฟอร์ของ BIG กลับมาไว้ในคีย์บอร์ดบัฟเฟอร์ ทันทีที่คีย์บอร์ดบัฟเฟอร์ว่างแล้วนำคาแรคเตอร์ทั้งหมดในบัฟเฟอร์ของตัวเองหลังจากย้ายคาแรคเตอร์ออกไปแล้วมาแสดงบนจอภาพด้วยเช่นกัน

สรุป

หลังจากโหลดโปรแกรม BIG.COM แล้วขนาดของคีย์บอร์ดบัฟเฟอร์ของคุณก็จะถูกขยายให้ใหญ่ขึ้น นอกจากนี้จะนำคาแรคเตอร์ในบัฟเฟอร์ออกแสดงบนจอภาพ และคุณยังสามารถใช้คีย์ BackSpace ลบคาแรคเตอร์ออกจากบัฟเฟอร์ได้เช่นเดิม

รูปที่ 1 โปรแกรม BIG.ASM เพิ่มบัฟเฟอร์ของคีย์บอร์ด

```
 1 bios_data     segment at 40h
 2               org   1Ah
 3 head          dw    ?
 4 tail          dw    ?
 5 buffer        dw    16 dup (?)
 6 buffer_end    label word
 7 bios_data     ends
 8 ;
 9 code          segment
10               assume cs:code
11               org   100h
12 start:        jmp   initialize
13 ;
14 buf           dw    0                   ;store a key that head point
15 bigbuf        dw    79 dup (0)          ;internal buffer of big_buffer
16 old_head      dw    0                   ;check for type ahead
17 display_on    db    0                   ;0=off, -1=on
18 old_int9h     label dword
19 old_kbd_int   dw    2 dup (0)           ;save old keyborad interrupt
20 old_int1Ch    label dword
21 rom_timer     dw    2 dup (0)           ;save old timer interrupt
22 video_seg     dw    0B800h              ;default video segment
23 ibm           db    'IBM'               ;EGA ASCII signature
24 adapter       db    2                   ;0 = MBA, 1 = CGA, 2 = EGA
25 enable_value  db    2Dh,28h,2Dh,29h,2Ah,2Eh,1Eh
26 ;
27 old_row       dw    80 dup (0)
28 screen_col1   dw    0720h
29 screen_row    dw    79 dup (0720h)
30 ;
31 msg           db    'BIG utility Copyright (C) 1989 (KJ+?''s Software)',13,10,'$'
32 errmsg        db    'BIG utility already installed.',13,10,'$'
33 ;
34 ;-------------------------------------------------------------------------
35 ; New interrupt 9 handler. Perform extend the keyboard buffer
36 ;-------------------------------------------------------------------------
37 big_buffer    proc  near
38               assume cs:code            ;all data reference use code segment
39               push  ax                  ;save all register
40               push  bx
41               push  cx
42               push  dx
43               push  si
44               push  di
45               push  ds
46               push  es
47               pushf                     ;save flag for IRET
48               call  old_int9h           ;call the old int 9h
49               assume ds:bios_data
50               mov   ax,bios_data
51               mov   ds,ax               ;DS point to bios data
52               mov   bx,tail             ;get current tail pointer
53               cmp   bx,head             ;is head equal to tail?
54               jne   big1                ;no, then work BIG_BUFFER
55               jmp   big_exit            ;yes, buffer is empty then exit
56 big1:         mov   bx,tail             ;read a char and head advance
57               sub   bx,2                ;point to read in character
58               cmp   bx,offset buffer    ;did we undershoot buffer?
59               jae   big2                ;no, don't adjust to top
60               mov   bx,offset buffer_end ;yes, then move to buffer top
61               sub   bx,2                ;adjust it
62 big2:         mov   bx,bx               ;use BX for index to a characterin buffer
63               mov   cx,[bx]             ;get key in CX
64               cmp   cx,buf              ;is it where we were before?
65               jne   big3                ;no
```

รูปที่ 1 (ต่อ)

```
66              mov   bx,head               ;has the head moved?
67              cmp   bx,old_head
68              je    big4                  ;if yes, we have read
69 big3:        cmp   bigbuf,0              ;is there anything in BIGBUF
70              jne   big5                  ;yes, BIGBUF isn't empty
71 big4:        cmp   dx,head               ;remove char in kbd buffer
72              jne   big5
73              jmp   big_exit              ;do nothing this pass
74 big5:        mov   bx,dx
75              mov   tail,dx               ;remove char by adjust tail
76              mov   dx,[bx]               ;store char in buffer
77              mov   cx,80                 ;count legnth
78              mov   bx,0                  ;find end of BIGBUF
79 big5a:       cmp   bigbuf[bx],0          ;do we meet the edn of BIGBUF?
80              jz    big6                  ;yes
81              inc   bx                    ;next word
82              inc   bx
83              loop  big5a
84              cmp   dx,0E08h              ;was this key a rub out?
85              jne   big_exit              ;no, and buffer filled -- leave
86              mov   bx,158                ;yes, buf full but rubout last char
87              mov   word ptr buf[bx],0
88              mov   bx,head
89              mov   old_head,bx           ;store this for next time
90              mov   dx,[bx]               ;always load BUFF
91              mov   buf,dx
92              jmp   big_exit              ;can't hold more than 80
93 big6:        cmp   dx,0E08h              ;rubout (and buffer not full)?
94              jne   big8                  ;no, don't del
95              sub   bx,2
96              cmp   bx,0FFFEh             ;gone too far?
97              jl    big_exit
98              jne   big7
99              mov   cx,tail               ;del the one char in kbd buffer
100             mov   head,cx               ;by making tail=head
101             mov   buf,0
102             jmp   short big9
103 big7:       mov   dx,0                  ;DX=0 if weare deleting
104 big8:       mov   bigbuf[bx],dx         ;load key in visitype buffers
105             mov   ax,dx
106             mov   ah,0Fh                ;intense attribute.
107             mov   screen_row[bx],ax     ;store to screen buffer
108             mov   bx,head
109             mov   old_head,bx           ;and store the old head to check later
110             mov   dx,[bx]               ;always reload BUF
111             mov   buf,dx
112             mov   dh,0Fh
113             mov   screen_col1,dx
114 big9:       cmp   display_on,0          ;are we on?
115             jne   big10                 ;yes, call display
116             mov   display_on,0FFh       ;store what's on the screen first
117             call  save_row
118 big10:      lea   si,screen_col1
119             call  fill_row
120 big_exit:   pop   es                    ;restore all register
121             pop   ds
122             pop   di
123             pop   si
124             pop   dx
125             pop   cx
126             pop   bx
127             pop   ax
128             iret
129 big_buffer  endp
130 ;
131 ;-----------------------------------------------------------------------
132 ; DISABLE_CGA and ENABLE_CGA control CGA video output.
133 ;-----------------------------------------------------------------------
134 disable_cga  proc  near
135             mov   dx,3DAh               ;status register port address
136 disable1:   in    al,dx                 ;read status
137             test  al,8                  ;vertical retrace active?
138             jz    disable1              ;no, wait until it is.
139             sub   dx,2                  ;point DX to MSR
140             mov   al,25h                ;load disable value
141             out   dx,al                 ;dsiable video
142             ret
143 disable_cga endp
144 ;
145 enable_cga   proc  near
146             mov   ah,15                 ;get current video mode
147             int   10h
148             lea   bx,enable_value       ;point BX to table of values
149             xlat  enable_value          ;get value to enable signal
150             mov   dx,3D8h               ;MSR address
151             out   dx,al                 ;enable video output
152             ret
153 enable_cga   endp
154 ;
155 ;-----------------------------------------------------------------------
156 ; SAVE_ROW saves the first row on screen.
157 ;-----------------------------------------------------------------------
158 save_row     proc  near
159             push  ds
160             push  di
161             push  si
162             push  cx
163             mov   ds,video_seg
164             assume ds:nothing
165             xor   si,si
166             lea   di,old_row
167             mov   cx,80
168             cmp   adapter,1
169             jne   save1
170             call  disable_cga
171 save1:      cld
172             rep   movsw
173             cmp   adapter,1
174             jne   save2
175             call  enable_cga
176 save2:      pop   cx
177             pop   si
178             pop   di
179             pop   ds
180             assume ds:code
181             ret
182 save_row     endp
183 ;
184 ;-----------------------------------------------------------------------
185 ; FILL_ROW writes the first row from the internal buffer.
186 ; Entry :  SI - address of buffer
187 ;-----------------------------------------------------------------------
188 fill_row      proc  near
189             push  es
190             push  di
191             push  cx
192             assume ds:nothing
193             push  ds
194             push  cs
195             pop   ds
196             mov   es,video_seg
197             xor   di,di
198             mov   cx,80
199             cmp   adapter,1
200             jne   fill1
201             call  disable_cga
202 fill1:      cld
203             rep   movsw
204             cmp   adapter,1
205             jne   fill2
206             call  enable_cga
```

รูปที่ 1 (ต่อ)

```
fill2:      pop   ds
            pop   cx
            pop   di
            pop   es
            assume ds:code
            ret
fill_row    endp
;
;-----------------------------------------------------------------------
; New interrupt 1Ch handler performs sliding a character from
; the internal buffer.
;-----------------------------------------------------------------------
timer       proc  near
            assume ds:code
            push  ds
            push  cs
            pop   ds                    ;point DS to current code segment
            pushf                       ;save flag for IRET
            call  old_int1Ch            ;call the old timer interrupt
            cmp   bigbuf,0              ;is there anything in BIGBUF?
            jne   timer1                ;yes, continue
            jmp   timer_exit            ;no, exit
timer1:     cli                         ;disable interrupt
            push  es                    ;save all register
            push  ds
            push  si
            push  di
            push  dx
            push  cx
            push  bx
            push  ax
            assume ds:bios_data
            mov   ax,bios_data
            mov   ds,ax                 ;point DS to bios data area
            mov   bx,tail
            cmp   head,bx               ;is the buffer empty?
            jne   timer4                ;no, then restore 1st row
            mov   ax,word ptr bigbuf    ;get a stored key from BIGBUF
            mov   cx,79                 ;slide the rest to the top
            mov   bx,0
timer2:     mov   dx,buf[bx+2]
            mov   buf[bx],dx
            mov   dx,screen_col1[bx+2]
            mov   screen_col1[bx],dx
            inc   bx
            inc   bx
            loop  timer2
            mov   word ptr bigbuf[bx-2],0     ;free the bottom
            mov   word ptr screen_row[bx-2],0720h
            mov   dx,head               ;save old head pointer
            mov   old_head,dx
            mov   dx,tail
            inc   dx                    ;adjust tail to next position for this wor
            inc   dx
            cmp   dx,offset buffer_end  ;are we past the end of buffer?
            jl    timer3                ;no, do not wrap
            mov   dx,offset buffer      ;wrap it
timer3:     cmp   dx,head               ;is buffer full?
            je    timer4                ;yes, leave and come back later
            mov   bx,tail               ;prepare to put character at tail
            mov   [bx],ax               ;put in buffer
            mov   tail,dx               ;adjust buffer tail
timer4:     cmp   bigbuf,0              ;should we restore the screen?
            jne   timer5                ;no
            mov   buf,0                 ;clear for restore
            mov   screen_col1,0720h
            mov   display_on,0
            lea   si,old_row
            call  fill_row
            jmp   short timer6
timer5:     lea   si,screen_col1
            call  fill_row
```

```
timer6:     pop   ax                    ;restore all register
            pop   bx
            pop   cx
            pop   dx
            pop   di
            pop   si
            pop   ds
            pop   es
timer_exit: pop   ds
            iret
timer       endp
;
;-----------------------------------------------------------------------
; INITIALIZE prepares the body of the program
;-----------------------------------------------------------------------
initialize  proc  near
            mov   ax,3509h
            int   21h
            cmp   bx,offset big_buffer
            jne   init0
            mov   dx,offset errmsg      ;display already load
            mov   ah,9
            int   21h
            mov   ax,4C01h
            int   21h
init0:      mov   ax,0C000h             ;set ES to EGA BIOS segment
            mov   es,ax
            mov   di,1Eh                ;point DI to signature location
            lea   si,iba                ;point SI to 'IBM' text
            mov   cx,3                  ;three bytes compare
            cld                         ;clear DF
            repe  cmpsb                 ;check three bytes
            je    init1                 ;we found the signature
            dec   adapter
            mov   ah,15                 ;get current video mode
            int   10h
            cmp   al,7                  ;is it mode 7?
            jne   init1                 ;no, it is CGA
            dec   adapter               ;set ADAPTER for MGA
            sub   video_seg,800h        ;modify video segment
init1:      assume ds:code
            mov   ax,3509h              ;read vector 9 addr.
            int   21h
            mov   old_kbd_int,bx
            mov   old_kbd_int[2],es     ;save it
            mov   dx,offset big_buffer  ;and change to BIG_BUFFER
            mov   ax,2509h
            int   21h
            mov   ax,351Ch              ;read vector 1Ch addr.
            int   21h
            mov   rom_timer,bx
            mov   rom_timer[2],es       ;save it
            mov   dx,offset timer       ;and change to TIMER
            mov   ax,251Ch
            int   21h
            assume ds:bios_data
            mov   ax,bios_data
            mov   ds,ax
            mov   bx,offset buffer
            mov   head,bx               ;reset keyboard buffer
            mov   old_head,bx
            mov   tail,bx
            push  cs
            pop   ds                    ;point DS to data
            mov   dx,offset msg         ;banner message
            mov   ah,9
            int   21h
            mov   dx,offset initialize  ;terminate but stay resident
            int   27h
initialize  endp
code        ends
            end   start
```